(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 May 2004 (13.05.2004)

PCT

(10) International Publication Number
WO 2004/040459 A1

(51) International Patent Classification[7]: G06F 12/02

(21) International Application Number:
PCT/US2003/028429

(22) International Filing Date:
10 September 2003 (10.09.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/281,824 28 October 2002 (28.10.2002) US

(71) Applicant (*for all designated States except US*): SANDISK CORPORATION [US/US]; 140 Caspian Court, Sunnyvale, CA 94089 (US).

(72) Inventors; and

(75) Inventors/Applicants (*for US only*): CHANG, Robert, C. [US/US]; 10 Stanton Court, Danville, CA 94506 (US). QAWAMI, Bahman [US/US]; 5899 killarney Circle, San Jose, CA 95138 (US). SABET-SHARGHI, Farshid [US/US]; 5634 Snowdon Place, San Jose, CA 95138 (US).

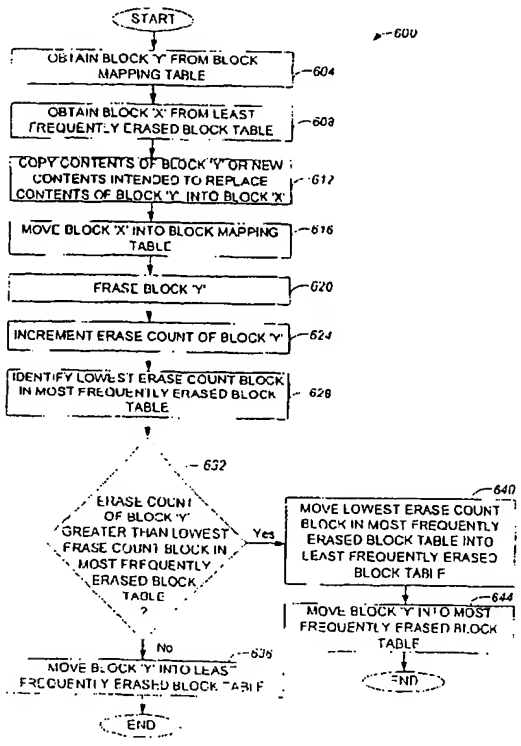
(74) Agent: SU, Peggy, A.; Ritter, Lang & Kaplan LLP, 12930 Saratoga Avenue, Suite D1, Saratoga, CA 95070 (US).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO,

[Continued on next page]

(54) Title: TRACKING THE LEAST FREQUENTLY ERASED BLOCKS IN NON-VOLATILE MEMORY SYSTEMS



(57) Abstract: Method and apparatus for performing wear leveling in a non-volatile memory system are disclosed. According to one aspect of the present invention, one method for processing elements included in a non-volatile memory of a memory system includes obtaining erase counts associated with elements and grouping a number of the elements into a first set. Each element has an associated erase count that substantially indicates a number of times the element has been erased. Grouping the number of elements into the first set includes selecting elements included in the plurality of elements which have the lowest associated erase counts of the erase counts associated with the plurality of elements. The method also includes storing the erase counts associated with the first set in a memory component substantially within a table.

WO 2004/040459 A1

SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declaration under Rule 4.17:

— *of inventorship (Rule 4.17(iv)) for US only*

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

TRACKING THE LEAST FREQUENTLY ERASED BLOCKS IN NON-VOLATILE MEMORY SYSTEMS

CROSS REFERENCE TO RELATED APPLICATIONS

The present invention is related to co-pending U.S. Patent Application No. _____ (Atty. Docket No. SANDP005/SDK0278.000US) entitled "AUTOMATED WEAR LEVELING IN NON-VOLATILE STORAGE SYSTEMS", co-pending U.S. Patent Application No. 10/281,739 (Atty. Docket No. SANDP023/SDK0366.000US) entitled "WEAR-LEVELING IN NON-VOLATILE SYSTEMS", filed October 28, 2002, co-pending U.S. Patent Application No. 10/281,670 (Atty. Docket No. SANDP025/SDK0366.002) entitled "TRACKING THE MOST FREQUENTLY ERASED BLOCKS IN NON-VOLATILE MEMORY SYSTEMS, filed October 28, 2002,, co-pending U.S. Patent Application No. 10/281,631 (Atty. Docket No. SANDP028/SDK0371.000US) entitled "METHOD AND APPARATUS FOR SPLITTING A LOGICAL BLOCK, filed October 28, 2002,, co-pending U.S. Patent Application No. 10/281,855 (Atty. Docket No. SANDP029/DSK0410.000US) entitled "METHOD AND APPARATUS FOR GROUPING PAGES WITHIN A BLOCK," filed October 28, 2002, co-pending U.S. Patent Application No. 10/281,762 (Atty. Docket No. SANDP030/SDK0416.000US) entitled "METHOD AND APPARATUS FOR RESOLVING PHYSICAL BLOCKS ASSOCIATED WITH A COMMON LOGICAL BLOCK," filed October 28, 2002 U.S. Patent No. 6,081,447, and U.S. Patent No. 6,230,233, which are each incorporated herein by reference in their entireties.

BACKGROUND OF THE INVENTION

1. Field of Invention

The present invention relates generally to mass digital data storage systems. More particularly, the present invention relates to systems and methods for allowing the wear associated with storage areas in a non-volatile storage system to be spread out across substantially all storage areas.

2. Description of the Related Art

The use of non-volatile memory systems such as flash memory storage systems is increasing due to the compact physical size of such memory systems, and the ability for non-volatile memory to be repetitively reprogrammed. The compact physical size of flash memory storage systems facilitates the use of such storage systems in devices which are becoming increasingly prevalent. Devices which use flash memory storage systems include, but are not limited to, digital cameras, digital camcorders, digital music players, handheld personal computers, and global positioning devices. The ability to repetitively reprogram non-volatile

BNSDOCID: <WO 2004040459A1_I_>

memory included in flash memory storage systems enables flash memory storage systems to be used and reused.

In general, flash memory storage systems may include flash memory cards and flash memory chip sets. Flash memory chip sets generally include flash memory components and a controller components. Typically, a flash memory chip set may be arranged to be assembled into an embedded system. The manufacturers of such assemblies or host systems typically acquire flash memory in component-form, as well as other components, then assemble the flash memory and the other components into a host system.

Although non-volatile memory or, more specifically, flash memory storage blocks within flash memory systems may be repetitively programmed and erased, each block or physical location may only be erased a certain number of times before the block wears out, *i.e.*, before memory begins to become smaller. That is, each block has a program and erase cycle limit. In some memory, a block may be erased up to approximately ten thousand times before the block is considered to be unusable. In other memory, a block may be erased up to approximately one hundred thousand times or even up to a million times before the block is considered to be worn out. When a block is worn out, thereby causing a loss of use or a significant degradation of performance to a portion of the overall storage volume of the flash memory system, a user of the flash memory system may be adversely affected, as for the example through the loss of stored data or the inability to store data.

The wear on blocks, or physical locations, within a flash memory system varies depending upon how much each of the blocks is programmed. If a block or, more generally, a storage element, is programmed once, then effectively never reprogrammed, the number of program and erase cycles and, hence, wear associated with that block will generally be relatively low. However, if a block is repetitively written to and erased, *e.g.*, cycled, the wear associated with that block will generally be relatively high. As logical block addresses (LBAs) are used by hosts, *e.g.*, systems which access or use a flash memory system, to access data stored in a flash memory system, if a host repeatedly uses the same LBAs to write and overwrite data, the same physical locations or blocks within the flash memory system are repeatedly written to and erased, as will be appreciated by those of skill in the art.

When some blocks are effectively worn out while other blocks are relatively unworn, the existence of the worn out blocks generally compromises the overall performance of the flash memory system. In addition to degradation of performance associated with worn out blocks themselves, the overall performance of the flash memory system may be compromised when an insufficient number of blocks which are not worn out are available to store desired data. Often, a flash memory system may be deemed unusable when a critical number worn out blocks are present in the flash memory system, even when many other cells in the flash memory system are

BNSDOCID: <WO 2004040459A1_I >

relatively unworn. When a flash memory system which includes a substantial number of relatively unworn blocks is considered to be unusable, many resources associated with the flash memory system are effectively wasted.

In order to increase the likelihood that blocks within a flash memory system are worn fairly evenly, wear leveling operations are often performed. Wear leveling operations, as will be understood by those skilled in the art, are generally arranged to allow the physical locations or blocks which are associated with particular LBAs to be changed such that the same LBAs are not always associated with the same physical locations or blocks. By changing the block associations of LBAs, it is less likely that a particular block may wear out well before other blocks wear out.

One conventional wear leveling process involves swapping physical locations to which two relatively large portions of customer or host LBAs are mapped. That is, the LBAs associated with relatively large sections of storage cells are swapped. Such swapping is initiated through a manual command from a customer, *e.g.*, through the use of a host and, as a result, is not transparent to the customer. Also, swapping operations that involve moving data between two relatively large sections of storage cells are time consuming and, hence, inefficient. Additionally, the performance of the overall flash memory system may be adversely affected by swapping operations of a relatively long duration which consume significant resources associated with the overall flash memory system. As will be appreciated by those skilled in the art, moving data from a first location typically involves copying the data into another location and erasing the data from the first location.

Another conventional wear leveling process involves allowing blocks to wear. Once the blocks have effectively worn out, the sectors assigned to the blocks may be reassigned by mapping the addresses associated with the sectors to spare areas once the blocks in which the sectors have been stored have worn out, or have become unusable. As the number of spare areas or blocks is limited and valuable, there may not always be spare areas to which sectors associated with unusable blocks may be mapped. In addition, effectively remapping sectors only after blocks have become unusable generally allows performance of the overall flash memory system to degrade.

Therefore, what are desired are a method and an apparatus for efficiently and substantially transparently performing wear leveling within a flash memory storage system. That is, what is needed is a wear leveling process which promotes more even wear in physical locations associated with the flash memory storage system without requiring a significant use of computational resources.

BNSDOCID: <WO 2004040459A1 >

SUMMARY OF THE INVENTION

The present invention relates to a system and a method for performing wear leveling in a non-volatile memory system. According to one aspect of the present invention, one method for processing elements included in a non-volatile memory of a memory system includes obtaining erase counts associated with elements and grouping a number of the elements into a first set. Each element has an associated erase count that substantially indicates a number of times the element has been erased. Grouping the number of erased elements into the first set includes selecting erased elements included in the plurality of elements which have the lowest associated erase counts of the erase counts associated with the plurality of elements. The method also includes storing the erase counts associated with the first set in a memory component substantially within a table.

In one embodiment, the elements in the first set are sorted according to the erase count associated with each element. In such an embodiment, the method may also include identifying an erased element included in the first set that has the lowest erase count associated with the first set, disassociating the element from the first set, and associating a different erased element of the plurality of elements with the first set.

By maintaining worn elements, *e.g.*, extended blocks, of a non-volatile memory in a group of worn elements, the wear of the worn elements may be managed essentially by preventing the worn elements from being allocated for use until less worn elements eventually become more worn. As such, the wear of all elements may be evened out, as the elements which have been erased more often are effectively prevented from being used and, hence, erased again, until other elements have been used and erased more. The overall life of a media, as for example a device which includes an embedded flash memory or a flash memory card, may generally be extended if blocks which have relatively low erase counts are used as spare blocks when a block containing not-needed data is erased, or when a block is needed to replace a substantially worn block. Maintaining spare blocks in a group of least erased blocks which are effectively tracked using a table stored in system memory facilitates the allocation of less worn blocks over blocks which are more worn.

According to another aspect of the present invention, a memory system includes a first memory that stores a table arranged to include entries associated with a first set of erased storage elements which have associated erase counters that are less than an average erase count. The associated erase counter for each storage element of the first set of storage elements is arranged to substantially indicate a number of times the storage element has been erased. The memory system also includes a second memory that includes a plurality of storage elements which includes the first set of storage elements. The average erase count is determined using erase counters associated with the plurality of storage elements. Finally, the memory system

BNSDOCID: <WO ...2004040459A1_I_>

includes a processor that may access the first memory and the second memory. In one embodiment, the first set of storage elements may be identified using the table.

In accordance with yet another aspect of the present invention, a method for managing memory that includes a plurality of blocks includes identifying a set of erased blocks included in the plurality of blocks, and identifying a first group of erased blocks included in the set of erased blocks. The first group of erased blocks includes erased blocks with lower erase counts than substantially all other erased blocks in the set of erased blocks. A structure which includes entries is created in a memory component. The entries include erase counts of the erased blocks included in the first group. The entries are sorted, and a first block is identified using the sorted entries. The first block is arranged to be removed from the first group of erased blocks prior to the other erased blocks included in the first group of erased blocks. In other words, the first block may be the first block swapped out of the first group on an as-needed basis.

In one embodiment, sorting the entries includes sorting the entries using the erase counts of the erased blocks included in the first group. In such an embodiment, identifying the first block using the sorted entries may includes identifying the first block to have a lower erase count than substantially all other erased blocks included in the first group.

These and other advantages of the present invention will become apparent upon reading the following detailed descriptions and studying the various figures of the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

Fig. 1a is a diagrammatic representation of a general host system which includes a non-volatile memory device in accordance with an embodiment of the present invention.

Fig. 1b is a diagrammatic representation a memory device, *e.g.*, memory device 120 of Fig. 1a, in accordance with an embodiment of the present invention.

Fig. 2 is a diagrammatic representation of a portion of a flash memory in accordance with an embodiment of the present invention.

Fig. 3 is a process flow diagram which illustrates the steps associated with processing an initialization request with respect to a flash memory system, in accordance with an embodiment of the present invention.

Fig. 4 is a process flow diagram which illustrates the steps associated with one method of processing a static block in accordance with an embodiment of the present invention.

Fig. 5a is a diagrammatic block diagram representation of a system memory in accordance with an embodiment of the present invention.

BNSDOCID <WO 2004040459A1_I_>

Fig. 5b is a diagrammatic representation of normal blocks, least frequently erased blocks, and most frequently erased blocks in accordance with an embodiment of the present invention.

Fig. 6 is a diagrammatic representation of one method of performing a block swap or update in the system memory an overall memory system to allow for more even wear of the blocks in accordance with an embodiment of the present invention.

Fig. 7 is a diagrammatic block diagram representation of a system architecture in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE EMBODIMENTS

Non-volatile memory storage blocks within flash memory storage systems may be repetitively programmed and erased, although each block may generally only be erased a finite number of times before the block wears out. When a block wears out, a relatively significant degradation of performance associated with the portion of the overall storage volume of the flash memory storage system that includes the worn out block occurs, and data stored in that portion may be lost, or it may become impossible to store data in that portion.

In order to increase the likelihood that blocks wear out more evenly within a flash memory storage system, blocks may be more evenly utilized. By keeping track of how many times each block has been erased, as for example through the utilization of an erase count, memory within a system may be more evenly used. An erase count management technique may store an erase count which keeps track of how many times a particular block has been erased in a redundant area associated with the block. Tables may be built in system memory which substantially enables blocks that are in use to effectively be separated from blocks which have relatively high erase counts and blocks which have relatively low erase counts. When a block in use is erased, the block may be "added" to either a table of blocks which have relatively high erase counts or a table of blocks which have relatively low erase counts, as appropriate. Likewise, blocks may be "moved" from either the table of blocks which have relatively high erase counts or the table of blocks which have relatively low erase counts into a block mapping table, *i.e.*, a set of table of blocks which are in use, to substantially replace any block which has been reassigned from the block mapping table.

By categorizing blocks, blocks may be more evenly utilized as the use of each block may be more effectively managed to even out the wear associated with the blocks. Further, categorizing blocks into tables enables blocks with a low erase count and blocks with a high erase count to be readily identified and, hence, does not utilize a significant amount of computational resources. Hence, wear leveling occurs relatively efficiently. As a result, the life of the flash memory system may be extended substantially without significantly affecting the performance of the flash memory system.

BNSDOCID: <WO_ _ 2004040459A1_I_>

Flash memory systems or, more generally, non-volatile memory devices generally include flash memory cards and chip sets. Typically, flash memory systems are used in conjunction with a host system such that the host system may write data to or read data from the flash memory systems. However, some flash memory systems include embedded flash memory and software which executes on a host to substantially act as a controller for the embedded flash memory. Referring initially to Fig. 1a, a general host system which includes a non-volatile memory device, *e.g.*, a CompactFlash memory card or an embedded system, will be described. A host or computer system 100 generally includes a system bus 104 which allows a microprocessor 108, a random access memory (RAM) 112, and input/output circuits 116 to communicate. It should be appreciated that host system 100 may generally include other components, *e.g.*, display devices and networking device, which are not shown for purposes of illustration.

In general, host system 100 may be capable of capturing information including, but not limited to, still image information, audio information, and video image information. Such information may be captured in real-time, and may be transmitted to host system 100 in a wireless manner. While host system 100 may be substantially any system, host system 100 is typically a system such as a digital camera, a video camera, a cellular communications device, an audio player, or a video player. It should be appreciated, however, that host system 100 may generally be substantially any system which stores data or information, and retrieves data or information.

It should be appreciated that host system 100 may also be a system which either only captures data, or only retrieves data. That is, host system 100 may be a dedicated system which stores data, or host system 100 may be a dedicated system which reads data. By way of example, host system 100 may be a memory writer which is arranged only to write or store data. Alternatively, host system 100 may be a device such as an MP3 player which is typically arranged to read or retrieve data, and not to capture data.

A non-volatile memory device 120 which, in one embodiment, is a removable non-volatile memory device, is arranged to interface with bus 104 to store information. An optional input/output circuit block 116 may allow non-volatile memory device 120 to interface indirectly with bus 104. When present, input/output circuit block 116 serves to reduce loading on bus 104, as will be understood by those skilled in the art. Non-volatile memory device 120 includes non-volatile memory 124 and an optional memory control system 128. In one embodiment, non-volatile memory device 120 may be implemented on a single chip or a die. Alternatively, non-volatile memory device 120 may be implemented on a multi-chip module, or on multiple discrete components which may form a chip set and may be used together as non-volatile memory device

BNSDOCID. <WO 2004040459A1_I_>

120. One embodiment of non-volatile memory device 120 will be described below in more detail with respect to Fig. 1b.

Non-volatile memory 124, *e.g.*, flash memory such as NAND flash memory, is arranged to store data such that data may be accessed and read as needed. Data stored in non-volatile memory 124 may also be erased as appropriate, although it should be understood that some data in non-volatile memory 124 may not be erasable. The processes of storing data, reading data, and erasing data are generally controlled by memory control system 128 or, when memory control system 128 is not present, by software executed by microprocessor 108. The operation of non-volatile memory 124 may be managed such that the lifetime of non-volatile memory 124 is substantially maximized by essentially causing sections of non-volatile memory 124 to be worn out substantially equally.

Non-volatile memory device 120 has generally been described as including an optional memory control system 128, *i.e.*, a controller. Often, non-volatile memory device 120 may include separate chips for non-volatile memory 124 and memory control system 128, *i.e.*, controller, functions. By way of example, while non-volatile memory devices including, but not limited to, PC cards, CompactFlash cards, MultiMedia cards, and Secure Digital cards include controllers which may be implemented on a separate chip, other non-volatile memory devices may not include controllers that are implemented on a separate chip. In an embodiment in which non-volatile memory device 120 does not include separate memory and controller chips, the memory and controller functions may be integrated into a single chip, as will be appreciated by those skilled in the art. Alternatively, the functionality of memory control system 128 may be provided by microprocessor 108, as for example in an embodiment in which non-volatile memory device 120 does not include memory controller 128, as discussed above.

With reference to Fig. 1b, non-volatile memory device 120 will be described in more detail in accordance with an embodiment of the present invention. As described above, non-volatile memory device 120 includes non-volatile memory 124 and may include memory control system 128. Memory 124 and control system 128, or controller, may be primary components of non-volatile memory device 120, although when memory 124 is an embedded NAND device, for example, non-volatile memory device 120 may not include control system 128. Memory 124 may be an array of memory cells formed on a semiconductor substrate, wherein one or more bits of data are stored in the individual memory cells by storing one of two or more levels of charge on individual storage elements of the memory cells. A non-volatile flash electrically erasable programmable read only memory (EEPROM) is an example of a common type of memory for such systems.

When present, control system 128 communicates over a bus 15 to a host computer or other system that is using the memory system to store data. Bus 15 is generally a part of bus 104

BNSDOCID: <WO 2004040459A1_1 >

of Fig. 1a. Control system 128 also controls operation of memory 124, which may include a memory cell array 11, to write data provided by the host, read data requested by the host and perform various housekeeping functions in operating memory 124. Control system 128 generally includes a general purpose microprocessor which has associated non-volatile software memory, various logic circuits, and the like. One or more state machines are often also included for controlling the performance of specific routines.

Memory cell array 11 is typically addressed by control system 128 or microprocessor 108 through address decoders 17. Decoders 17 apply the correct voltages to gate and bit lines of array 11 in order to program data to, read data from, or erase a group of memory cells being addressed by the control system 128. Additional circuits 19 include programming drivers that control voltages applied to elements of the array that depend upon the data being programmed into an addressed group of cells. Circuits 19 also include sense amplifiers and other circuits necessary to read data from an addressed group of memory cells. Data to be programmed into array 11, or data recently read from array 11, are typically stored in a buffer memory 21 within control system 128. Control system 128 also usually contains various registers for temporarily storing command and status data, and the like.

Array 11 is divided into a large number of BLOCKS 0 – N memory cells. As is common for flash EEPROM systems, the block is typically the smallest unit of erase. That is, each block contains the minimum number of memory cells that are erased together. Each block is typically divided into a number of pages, as also illustrated in Fig. 2. A page is typically the smallest unit of programming. That is, a basic programming operation writes data into or reads data from a minimum of one page of memory cells. One or more sectors of data are typically stored within each page. As shown in Fig. 1b, one sector includes user data and overhead data. Overhead data typically includes an error correction code (ECC) that has been calculated from the user data and overhead data of the page. A portion 23 of the control system 128 calculates the ECC when data is being programmed into array 11, and also checks the ECC when data is being read from array 11. Alternatively, the ECCs are stored in different pages, or different blocks, than the user data to which they pertain.

A sector of user data is typically 512 bytes, corresponding to the size of a sector in magnetic disk drives. Overhead data is typically an additional 16 bytes. One sector of data is most commonly included in each page but two or more sectors may instead form a page. Any number of pages may generally form a block. By way of example, a block may be formed from eight pages up to 512, 1024 or more pages. The number of blocks is chosen to provide a desired data storage capacity for the memory system. Array 11 is typically divided into a few sub-arrays (not shown), each of which contains a proportion of the blocks, which operate somewhat independently of each other in order to increase the degree of parallelism in the execution of

BNSDOCID: <WO 2004040459A1_I_>

various memory operations. An example of the use of multiple sub-arrays is described in U.S. Patent No. 5,890,192, which is incorporated herein by reference in its entirety.

When a particular section, *e.g.*, storage element, of non-volatile memory 124 is programmed continuously, *e.g.*, written to and erased repeatedly, that particular area generally wears out more quickly than an area which is not programmed continuously. In order to effectively "even out" the wear of different areas within non-volatile memory 124, wear leveling may be substantially automatically performed such that areas which are programmed continuously are programmed less, while areas that are not programmed continuously may be programmed more.

Generally, to perform wear leveling, a block, *e.g.*, a set of sectors which are associated with a physical location, which is programmed repeatedly may be swapped with a block which is associated with a physical location which is not programmed repeatedly. That is, a physical block which has been programmed and, hence, erased repeatedly may be swapped with a physical block which has been programmed and erased less often.

In one embodiment of the present invention, in order for it to be readily determined whether a particular physical block has been programmed and erased repeatedly, an erase count may be stored with the block. That is, a counter which keeps track of how many times a block has been erased may be maintained and incremented each time the block is erased. Such an erase count may be used to facilitate a determination of whether a particular block should be swapped with another block which has been erased less often. Fig. 2 is a diagrammatic representation of a portion of a flash memory in accordance with an embodiment of the present invention. Flash memory 200 may be divided into pages 204. Each page 204, which generally contains approximately 512 bytes of user data, effectively includes a redundant area 206, *e.g.*, page 204a includes redundant area 206a. Each redundant area 206 or overhead area may include up to approximately sixteen bytes of information which typically includes, but is not limited to, a group identifier 216, an update index 212, and an erase count 214.

Typically, any number of pages 204 are included in a block 210. For ease of illustration, pages 204a, 204b are shown as being included in block 210, although it should be appreciated that the number of pages 204 included in block 210 may vary widely. In the described embodiment, block 210 may be arranged to include approximately 32 pages. For example, when flash memory 200 includes approximately 512 MegaBytes (MB), flash memory 200 may effectively be divided into approximately 4096 blocks of 32 pages each.

As previously mentioned, erase count 214 may be incremented each time user data is erased from an associated block. For instance, erase count 214, which is associated with block 210, may be incremented each time data is erased from block 210. Since each page 204a, 204b

BNSDOCID: <WO__ __2004040459A1_I_>

included in block 210 generally has an erase count 214, the erase count 214 associated with each page 204a, 204b may be incremented when block 210 is erased.

When a block containing data is erased, both the data area and the redundant area are typically erased or emptied. The erased block is added to spare block pools, which contains the erased blocks containing smaller erase count comparing to erased blocks of other tables. The spare block table is essentially a least frequently erased bloc table. In one embodiment of the present invention, an erased block which has a relatively large erase count is added to the pool containing the erased blocks which have larger erase counts than the erased blocks contained in other tables. That pool, *i.e.*, the pool containing the erased blocks which have larger erase counts, is the most frequently erased block table. The erase count of a just-erased block is incremented by one and is saved in the least frequently erased block table or the most frequently erased block table depending on the value of the erase count.

An erase count such as erase count 214 may be accessed during an initialization request. An initialization request may be made, for example, when a system, *e.g.*, a system which includes embedded flash memory, is powered up, when spare blocks within a system are running low, when a user makes a request to balance block allocation, and when a user makes a request for block usage to occur more evenly. Fig. 3 is a process flow diagram which illustrates the steps associated with processing an initialization request with respect to a flash memory system, in accordance with an embodiment of the present invention. In general, an initialization request may either be initiated by a user or substantially automatically initiated by a controller associated with flash memory system, *e.g.*, periodically or when a triggering condition is met. A process 300 of responding to an initialization request begins at step 304 in which an initialization request is effectively received. An initialization request may be received by a controller or a processor which is in communication with flash memory which is to be initialized. Such a request may be provided by a user via a host at power up, or when block allocation is to be balanced, for example.

Once the initialization request is received, an average erase count is obtained in step 306. In one embodiment, the average erase count is stored in an erase count block which is written into NAND memory associated with the system. The erase count block containing the average erase count and the erase count of each block is stored in a block of the flash memory. It should be appreciated that when an erase count block is created, *e.g.*, when the system is initially formatted, the average erase count and the erased count of each block are typically initialized to a value of zero. After the average erase count is obtained, erase counts for substantially all blocks within the system are obtained. As described above with respect to Fig. 2, the erase count for a particular block containing data is stored in a redundant area that is associated with that block. Hence, obtaining the erase count for substantially all blocks containing data may include

BNSDOCID: <WO 2004040459A1 I >

accessing a redundant area associated with each block, and storing each erase count into the erase count block at an initialization request.

After the initialization request is completed, the erase count of the erased block may not necessarily be updated in the erase count block. The erase count block generally retains its value because the redundant area of the erased block is erased. When the overall system is shut down, a termination request is made so that the erase count block is updated to contain the latest erased count of substantially all blocks. The erase count of an erased block belonging to a least frequently erased block is retrieved from the least frequently erased block table. The erase count of an erased block belonging to a most frequently erased block is retrieved from the most frequently erased block table. The erase count of substantially any remaining erased block may be retrieved from the erase count block.

In step 320, a block mapping table is allocated in the NAND memory. As will be appreciated by those skilled in the art, a block mapping table may be arranged to provide a mapping between a logical block address (LBA) and a physical block address (PBA). Additionally, a most frequently erased block table and a least frequently erased block table are also allocated in step 320.

A most frequently erased block table is typically sized or otherwise configured to effectively hold information relating to erased blocks which have been erased most frequently. That is, a most frequently erased block is arranged to hold information, *e.g.*, erase counts and mapping information, pertaining to erased blocks with the highest erase counts in the system. Similarly, a least frequently erased block table is generally sized or otherwise configured to accommodate information pertaining to erased blocks with the lowest erase counts. Although the size of the most frequently erased block table and the size of the least frequently erased block table may vary widely, the sizes are dependent upon the number of blocks which are to be designated as most frequently erased and the number of blocks which are to be designated as least frequently erased. Typically, the most frequently erased block table is generally sized to accommodate information for fewer erased blocks than the least frequently erased block table. By way of example, the least frequently erased block table may be sized to accommodate information for approximately seventy erased blocks, while the most frequently erased block table may be sized to accommodate information relating to approximately eighteen erased blocks. Alternatively, the least frequently erased block table may be sized to accommodate information for approximately fifty erased blocks, while the most frequently erased block table may be sized to accommodate information for approximately ten erased blocks.

After tables are allocated in step 320, erased blocks are identified in step 324. Then, in step 328, "N" erased blocks may be assigned to the most frequently erased blocks and essentially be assigned to the most frequently erased table. In one embodiment, the "N" erased blocks may

BNSDOCID: <WO 2004040459A1_I_>

be the "N" erased blocks with the highest erase counts as determined by a comparison of all erase counts. Alternatively, the "N" erased blocks to store in the most frequently erased block table may be determined based upon a comparison against the average erase count obtained in step 306. For instance, the "N" erased blocks may be "N" erased blocks which have an erase count that is at least a given percentage, *e.g.*, approximately twenty-five percent, higher than the average erase count.

Once the most frequently erased block table is effectively populated, "M" erased blocks may be identified and effectively be assigned to the least frequently erased block table in step 332. The "M " erased blocks may generally be the "M" erased blocks with the lowest erase counts of all erased blocks associated with the system, or the "M" erased blocks may be "M" erased blocks which have an erase count that is at least a given percentage lower than the average erase count. The "M" erased blocks are effectively spare blocks which will be assigned to the block mapping table as appropriate.

Remaining erased blocks, *i.e.*, erased blocks which have not be assigned to either the least frequently erased block table or the most frequently erased block table, are assigned to the block mapping table along with "unerased" blocks in step 336. In other words, remaining erased blocks as well as blocks containing data other than in associated redundant areas are associated with the block mapping table.

After the block mapping table, the least frequently erased block table, and the most frequently erased block table are effectively populated, *e.g.*, with erase counts and mapping information pertaining to corresponding blocks, an average erase count may be determined in step 338. Determining the average erase count typically involves summing the erase counts of individual blocks which were obtained in step 308, and dividing the sum by the total number of blocks.

The average erase count calculated in step 338 is stored into the erase count block associated with the system. As previously mentioned, the average erase count is stored in an erase count block which is written into NAND memory associated with the system. Upon storing the average erase count into the erase count block, static blocks, or blocks which contain data and have a relatively low associated erase count, may be processed in step 342. The steps associated with one method of processing a static block will be described below with respect to Fig. 4. Once the static blocks are processed, the process of processing an initialization request is completed.

Within a group of blocks associated with a flash memory, there are usually blocks which are erased and blocks which contain data, *i.e.*, user data, at any given time. Some of the blocks which contain data may be considered to be "normal" blocks, while others may be considered to be static blocks. Static blocks are blocks that contain data which is rarely changed. In other

BNSDOCID: <WO 2004040459A1_I_>

words, static blocks are rarely erased. Typically, static blocks may be associated with relatively old documents stored in flash memory, an executable program stored in the flash memory, or an operating system stored in the flash memory. A static block may generally have an erase count that is substantially lower than the erase count of the majority of blocks within flash memory. In one embodiment, a block which contains data may be considered to be a static block if the erase count of the block is below a certain percentage, *e.g.*, approximately twenty percent, of the average erase count associated with a flash memory system.

Since a static block contains data that is rarely changed, the data contained in the static block may be copied into a block which has a relatively high erase count. That is, when the contents of a particular physical block are relatively static and, hence, are generally not changed, the contents may effectively be reassigned to a different physical block which has a relatively high erase count in order to enable the original physical block, which has a relatively low erase count, to be used to store contents which are changed more frequently. With reference to Fig. 4, the steps associated with processing a static block, *i.e.*, step 342 of Fig. 3, will be described in accordance with an embodiment of the present invention. A process 342 of processing a static block of a system begins at step 404 in which the erase count of a block, *e.g.*, block "A," is accessed. Once the erase count of block "A" is accessed, a determination is made in step 408 regarding whether the erase count of block "A" is very low compared to the average erase count associated with the system.

Although a determination of whether the erase count of non-erased block "A" is low compared to the average erase count may be based on substantially any suitable criteria, in one embodiment, the determination is made based on whether the erase count of block "A" has a value that is less than a value associated with a fraction of the average erase count. For example, the erase count of block "A" may be considered to be low when the erase count is less than a predetermined percentage of the average erase count.

If it is determined in step 408 that the erase count of block "A" is not very low compared to the average erase count, then the indication is that block "A" is most likely not a static block. It should be appreciated that while block "A" may still be a static block even if the erase count of block "A" is not considered to be very low, the erase count of block "A" in such a case would effectively not trigger a swap of block "A" with another block. Accordingly, the processing of a static block is completed.

Alternatively, if it is determined in step 408 that the erase count of block "A" is very low compared to the average erase count, then the implication is that the contents of block "A" may be written into a block with a relatively high erase count such that block "A" with its low erase count may be free to store data that is changed relatively frequently. In other words, the indication when the erase count of block "A" is very low compared to the average erase count is

BNSDOCID <WO 2004040459A1_I_>

that block "A" is a static block. As such, process flow moves from step 408 to step 412 in which block "A" is identified as a static block. Once block "A" is identified as a static block, a block, namely block "B," may be obtained from a group of most frequently erased blocks as identified by the most frequently erased block table in step 416.

After block "B" is obtained, the contents of block "A" are copied into block "B" in step 420. That is, the user data contained in block "A" is copied into block "B" in step 420. Once the contents of block "A" are copied into block "B," block "A" is erased in step 424. Typically, when block "A" is erased, the erase count associated with block "A" is incremented. A block, *e.g.*, block "C," may be moved from the group of least frequently erased blocks into the group of most frequently erased blocks in step 428 such that the association of block "C" is effectively changed to the most frequently erased block table from the least frequently erased block table. In other words, block "C" is disassociated from the least frequently erased block table and associated with the most frequently erased block table. Such a move allows a space in the least frequently erased block table to effectively be opened up to accommodate block "A," which has a low erase count and, hence, is one of the least frequently erased blocks in the system. Typically, block "C" is the block with the highest erase count of the least frequently erased blocks.

Upon moving block "C" out of the group of least frequently erased blocks, or otherwise disassociating block "C" from the least frequently erased block table, process flow moves from step 428 to step 432 in which block "A" is effectively moved from the block mapping table into the least frequently erased block table in step 432. Then, in step 434, block "B," which includes contents that were previously contained in block "A," is associated with the block mapping table. As will be appreciated by those skilled in the art, "moving" block "B" into the block mapping table typically includes updating the mapping of a logical block address that was associated with block "A" to now be associated with block "B." When information pertaining to block "C" is present in the most frequently erased block table, information pertaining to block "B" is present in the block mapping table, and information pertaining to block "A" is present in the least frequently erased block table, the process of processing a static block is completed. It should be understood that process 342 may be repeated until substantially all static blocks associated with a system have been identified and processed.

In general, a block mapping table, a least frequently erased block table, and a most frequently erased block table may be created in system memory, *e.g.*, RAM 112 of Fig. 1a, when an initialization request is sent to an overall flash memory system. The tables may also be built substantially anytime an initialization request is made, as discussed above. To build the tables, space may first be allocated in system memory to accommodate the tables.

BNSDOCID: <WO 2004040459A1_I_>

As mentioned above, a block mapping table, a least frequently erased block table, and a most frequently erased block table are created in system memory, as is an average erase count. The average erase count and the erase count of each block are also written, in one embodiment, to the erase count block. Fig. 5a is a diagrammatic block diagram representation of a system memory in accordance with an embodiment of the present invention. A system memory 454 and a flash memory 460 are included in an overall system, and may, for example, effectively be components of a memory card or components of a host device in which flash memory 460 is embedded. System memory 454 is arranged to store a block mapping table 462 with which blocks may be associated. Typically, block mapping table 462 may be used in order to associate LBAs with physical blocks associated with flash memory 460.

System memory 454 also holds a least frequently erased block table 466 and a most frequently erased block table 470 which, like block mapping table 462, are generally formed in response to an initialization request. An average erase count 474, which is arranged to hold the average erase count of blocks within flash memory 460, is created when an overall flash memory system is formatted. In one embodiment, erase count block 480 is also arranged to contain the erase counts of substantially all blocks within flash memory 460. Each time an initialization request is made, an updated average erase count may be calculated, and stored into erase count block 480.

Fig. 5b is a diagrammatic representation of a group of "normal" blocks, a group of least frequently erased blocks, and a group of most frequently erased blocks in accordance with an embodiment of the present invention. A group of blocks 502 includes blocks 514 which may be normal or static blocks which generally contain user data, which may be erased but does not fall into the categories of a least frequently erased block or a most frequently erased block, as will be described below. A group least frequently erased blocks 506 generally includes blocks 518 which have the lowest erase counts within an overall system, while a group of most frequently erased blocks 510 generally includes blocks 522 which have the highest erase counts of the erased blocks within the overall system. In general, blocks 518 are effectively used as spare blocks.

When a block 514 is erased, it may be determined whether erased block 514 has a relatively low associated erase count or a relatively high associated erase count. When erased block 514 has a relatively low associated erase count, erased block 514 may be added to group of least frequently erased blocks 506. On the other hand, when erased block 514 has a relatively high associated erase count, erased block 514 may be reassigned to group of most frequently erased blocks 510.

Group of least frequently erased blocks 506, which may be of substantially any size, may be a sorted group. That is, blocks 518 may be substantially sorted based on erase counts. The

BNSDOCID: <WO 2004040459A1_I_>

sorting is typically reflected in a corresponding least frequently erased block table (not shown) which contains entries associated with blocks 518. For instance, each time a new block 518 is moved into or added to, or otherwise associated with, group of least frequently erased blocks 506, blocks 518 may essentially be sorted based on erase counts such that the least frequently erased block 518 in group of least frequently erased blocks 506 may be the next block 518 to be reassigned, as for example to group 502. In other words, when a new block into which data is to be copied is needed, the least erased block 518 of blocks 518 is identified using a least frequently erased block table, and taken from group of least frequently erased blocks 506. Typically, when a block 514 which contains data that is not needed is erased, that block 514 may be stored into group of least frequently erased blocks 506, and the least frequently erased block table may be updated accordingly, *i.e.*, an entry which corresponds to the added block may be included in the least frequently erased block table.

Blocks 522 in group of most frequently erased blocks 510, like blocks 518 stored in group of least frequently erased blocks 506, may also be substantially sorted based on erase counts. The sorting is typically implemented by sorting entries in a most frequently erased block table (not shown) which serves to identify blocks 522. In one embodiment, an average erase count associated with blocks 522 may be calculated, *i.e.*, an average erase count for group of most frequently erased blocks 510 may be determined. When a block 514 from group 502 is erased, and the erase count of the erased block 514 is found to exceed the average erase count for group of most frequently erased blocks 510 by more than a given percentage, *e.g.*, more than approximately twenty percent, the erased block 514 may be added to group of most frequently erased blocks 510. When a new block 522 is effectively added to group of most frequently erased blocks 510, a block 522 within group of frequently erased blocks 510 that has the lowest erase count may be reassigned into group 502. Such reassignments are typically reflected by updating an associated block mapping table, least frequently erased block table, and most frequently erased block table (not shown).

The swapping of blocks between group 502, group of least frequently erased blocks 506, and most frequently erased blocks 510 may generally occur when a block 514 included in group 502 is to be erased or updated. Alternatively, the swapping or updating of blocks may occur substantially any time it is desired for a spare block to be allocated for use in group 502.

Referring next to Fig. 6, one method of performing a block swap in an overall memory system such as a host system with embedded flash memory to allow for more even wear of the blocks will be described in accordance with an embodiment of the present invention. A process 600 of performing a block swap or update begins at step 604 in which a block, *e.g.*, block "Y," is "obtained" from a block mapping table or otherwise identified using the block mapping table.

BNSDOCID <WO 2004040459A1 I >

The block that is obtained is the block that is to be effectively swapped out of the block mapping table for copying or updating its contents.

Once block "Y" is obtained, a block, *e.g.*, block "X," is effectively obtained in step 608 from the least frequently erased block table. That is, a spare block is obtained from the group of least frequently erased blocks using the least frequently erased block table to identify an appropriate spare block. In general, block "X" is the block with the lowest erase count in the group of least frequently erased blocks, although it should be appreciated that block "X" may be substantially any block associated with the group of least frequently erased blocks and, hence, the least frequently erased block table. The contents or, more specifically, data contents stored in block "Y" or the new contents which are to replace the original contents of "Y" are copied into block "X" in step 612.

After the contents of block "Y" are copied into block "X," block "X" is effectively moved into, or associated with, the block mapping table in step 616. In other words, mappings associated with block "Y" and block "X" are effectively updated such that an LBA which was previously mapped to block "Y" is remapped to block "X." When block "X" is effectively moved into the block mapping table, block "Y" is erased in step 620. Specifically, the data contents, *e.g.*, user contents, stored in block "Y" may be erased using substantially any suitable technique. The erase count associated with block "Y," which is stored in a redundant area associated with block "Y," is then incremented in step 624 to indicate that block "Y" has once again been erased. It should be appreciated that in one embodiment, an erase count for "Y" which is effectively stored in an erase count block may be updated.

In step 628, the block with the lowest erase count in the most frequently erased block table is identified. As described above, in one embodiment, blocks referenced in the most frequently erased block table are sorted according to their respective erase counts. Sorting the blocks may include positioning the references to the blocks within the most frequently erased block table according to the erase counts of the blocks. Hence, identifying the block with the lowest erase count generally involves accessing the block reference in the position within the most frequently erased block table that is arranged to accommodate the block reference with the lowest erase count.

Once the block with the lowest erase count referenced in the most frequently erased block table is identified, process flow moves from step 628 to step 632 in which it is determined if the erase count of block "Y" is greater than the erase count of the block with the lowest erase count referenced in the most frequently erased block table. If it is determined that the erase count of block "Y" is not greater than the erase count of the block with the lowest erase count referenced in the most frequently erased block table, then the indication is that block "Y" is not considered to be frequently erased. Accordingly, process flow proceeds from step 632 to step 636 in which

BNSDOCID: <WO 2004040459A1 I>

block "Y" is moved into the group of least frequently erased blocks and effectively moved into the least frequently erased block table, *i.e.*, an entry corresponding to block "Y" is added into the least frequently erased block table. It should be appreciated that, in one embodiment, moving block "Y" into the group of least frequently erased blocks may include resorting substantially all block references in the least frequently erased block table using the erase count of each block. After block "Y" is effectively moved into the least frequently erased block table, the process of swapping or updating blocks is completed.

Returning to step 632, if the determination is step 632 is that the erase count of block "Y" exceeds the lowest erase count associated with the most frequently erased block table, the indication is that block "Y" should be moved into the group of most frequently erased blocks and effectively into the most frequently erased block table. In order for there to be room for block "Y" to be referenced in the most frequently erased block table, a block, *e.g.*, the block with the lowest erase count referenced in the most frequently erased block table, effectively needs to be removed from the most frequently erased block table. As such, in step 640, the block with the lowest erase count referenced in the most frequently erased block table is moved into the group of least frequently erased blocks, and effectively moved into the least frequently erased block table. Moving the block into the group of least frequently erased blocks may include resorting the block references in the least frequently erased block table according to the erase count of each block.

After the block with the lowest erase count in the most frequently erased block table is effectively moved out of the most frequently erased block table, block "Y" is effectively moved into the most frequently erased block table in step 644. In one embodiment, moving block "Y" into the group of most frequently erased blocks and, hence, effectively into the most frequently erased block table, may include resorting the most frequently erase blocks according to the erase count of each block, including block "Y." When block "Y" is effectively moved into the most frequently erased block table, the process of swapping blocks is completed.

In general, the functionality associated with maintaining tables, handling initialization requests, and performing wear leveling, *e.g.*, responding to requests to swap blocks, is provided in software, *e.g.*, as program code devices, or as firmware to a host system. One embodiment of a suitable system architecture associated with the software or firmware provided to a host system to enable wear leveling to occur is shown in Fig. 7. A system architecture 700 generally includes a variety of modules which may include, but are not limited to, an application interface module 704, a system manager module 708, a data manager module 712, a data integrity manager 716, and a device manager and interface module 720. In general, system architecture 700 may be implemented using software code devices or firmware which may be accessed by a processor, *e.g.*, processor 108 of Fig. 1a.

BNSDOCID: <WO 2004040459A1_I_>

In general, application interface module 704 may be arranged to communicate with the host, operating system or the user directly. Application interface module 704 is also in communication with system manager module 708 and data manager module 712. When the user wants to read, write or format a flash memory, the user sends requests to the operating system, the requests are passed to the application interface module 704. Application interface module 704 directs the requests to system manager module 708 or data manager module 712 depending on the requests.

System manager module 708 includes a system initialization submodule 724, an erase count block management submodule 726, and a power management block submodule 730. System initialization submodule 724 is generally arranged to enable an initialization request to be processed, and typically communicates with erase count block management submodule 726. In one embodiment, system initialization submodule 724 allows erase counts of blocks to be updated, and is substantially responsible for creating a least frequently erased block table and a most frequently erased block table.

Erase count block management submodule 726 includes functionality to cause erase counts of blocks to be stored, and functionality to cause an average erase count to be calculated, as well as updated, using individual erase counts. In other words, erase count block management submodule 726 effectively allows an average erase count to be maintained. Further, in one embodiment, erase count block management submodule 726 also substantially synchronizes the erase count of substantially all blocks in an erase count block during a power up of an overall system. While erase count block management submodule 726 may be arranged to cause an average erase count to be stored in an erase count block, it should be appreciated that power management block submodule 730 may instead be used to enable the average erase count to be stored.

In addition to being in communication with application interface module 704, system manager module 708 is also in communication with data manager module 712, as well as device manager and interface module 720. Data manager module 712, which communicates with both system manager module 708 and application interface module 704, may include functionality to provide page or block mapping. Data manager module 712 may also include functionality associated with operating system and file system interface layers.

Device manager and interface module 720, which is in communication with system manager module 708, data manager 712, and data integrity manager 716, typically provides a flash memory interface, and includes functionality associated with hardware abstractions, *e.g.*, an I/O interface. Data integrity manager module 716 provides ECC handling, among other functions.

BNSDOCID: <WO_ 2004040459A1 I >

Although only a few embodiments of the present invention have been described, it should be understood that the present invention may be embodied in many other specific forms without departing from the spirit or the scope of the present invention. By way of example, in lieu of assigning blocks to a most frequently erased block table and a least frequently erased block table based on a comparison of each block against an average erase count, blocks may instead be assigned to a most frequently erased block table and a least frequently erased block table based on a substantially absolute determination of which blocks have the highest erase counts and which blocks have the lowest erase counts, respectively. In other words, rather than comparing individual block erase counts against an average erase count, block erase counts may effectively be compared against each other to determine an appropriate table in which to insert a block.

A least frequently erased block table has generally been described as holding references to blocks with a relatively low erase count as spare blocks. Spare blocks are effectively allocated for use through the use of a block mapping table such that substantially any time a spare block is needed, the block with the lowest erase count referenced in the least frequently erased block table is provided for use. In other words, when a block identified in a block mapping table is to be swapped out, a reference to the block with the lowest erase count in the group of least frequently erased blocks is moved into the block mapping table. It should be appreciated, however, that substantially any block may generally be taken from the group of least frequently erased blocks during a block swapping or updating process. Selecting substantially any block from the least frequently erased blocks using the least frequently erased block table to move into the block mapping table may reduce the overhead associated with an overall system, as the blocks within the least frequently erased block table may not necessarily be sorted.

Identifying and processing static blocks generally enhances the ability to enable blocks within an overall memory system to be worn evenly. In one embodiment, however, static blocks are not necessarily identified and processed. For example, if a relatively low number of static blocks is anticipated within a system, the identification and processing of static blocks may be substantially eliminated without departing from the spirit or the scope of the present invention.

While non-volatile memory systems have been described as being controlled by associated memory controllers or being controlled using software or firmware associated with a host system, it should be understood that wear leveling processes which include erase count management may be applied to non-volatile memory systems which are in communication with controllers which are substantially external to the non-volatile memory systems. Suitable memory systems which use controllers include, but are not limited to, PC cards, CompactFlash cards, MultiMedia cards, Secure Digital cards, and embedded chip sets which include flash memory and a flash memory controller. Memory systems which are controlled through the use of software or firmware loaded onto a host system include embedded memory devices. In one

BNSDOCID <WO 2004040459A1_I_>

embodiment, memory systems which may use the erase management techniques described above and do not use c· trollers associated with the memory systems may use controllers associated with a host, *e.g.*, a host computer system, to implement wear leveling. That is, a host may directly address and manage memory in which wear leveling is to occur through the use of a controller on the host.

In general, the steps associated with the various processes and methods of wear leveling may vary widely. Steps may generally be added, removed, altered, and reordered without departing from the spirit of the scope of the present invention. By way of example, processing static blocks may not necessarily be included in the processing an initiation request. Also, in one embodiment, the determination of whether to effectively place a newly erased block into a most frequently erased block table may be based upon other criteria in lieu of being based upon whether the erased block has an erase count that is greater than the lowest erase count associated with the most frequently erased block table. For instance, such a determination may be based upon whether the erase count of the erased block exceeds an average erase count of substantially all blocks associated with the most frequently erased block table for a certain percentage, *e.g.*, approximately twenty percent. When the erase count of the erased block exceeds the average erase count by more than the certain percentage, then the block with the lowest erase count referenced in the most frequently erased block table may be moved into a least frequently erased block table, while the erased block is moved into the most frequently erased block table. Therefore, the present examples are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope of the appended claims.

BNSDOCID <WO 2004040459A1_I >

WHAT IS CLAIMED IS:

1. A method for processing elements included in a non-volatile memory of a memory system, the method comprising:

obtaining erase counts associated with a plurality of elements, wherein each element included in the plurality of elements has an associated erase count, the associated erase count of each element being arranged to indicate a number of times the element has been erased;

grouping a number of elements included in the plurality of elements into a first set, wherein grouping the number of erased elements into the first set includes selecting elements included in the plurality of elements which have the lowest associated erase counts of the erase counts associated with the plurality of erased elements; and

storing the erase counts associated with the first set in a memory component substantially within a table, wherein the memory component is associated with the memory system.

2. The method of claim 1 wherein grouping the number of elements includes:

comparing the erase counts associated with the plurality of elements; and

identifying a predetermined number of elements to be the number of elements selected from the plurality of erased elements which have the lowest associated erase counts of the erase counts associated with the plurality of erased elements.

3. The method of claim 1 further including:

sorting the elements in the first set according to the erase count associated with each element.

4. The method of claim 3 wherein sorting the elements in the first set includes sorting the erase counts within the table.

5. The method of claim 3 further including:

identifying a first element included in the first set, the first element having the lowest erase count associated with the first set.

6. The method of claim 5 further including:

disassociating the first element from the first set; and

associating a second element of the plurality of elements with the first set.

7. The method of claim 6 further including:

BNSDOCID: <WO 2004040459A1 I_>

disassociating the second element from the second set.

8. The method of claim 1, wherein the non-volatile memory is flash memory.

9. The method of claim 8 wherein the plurality of elements are blocks, and the first set is a set of least frequently erased blocks.

10. The method of claim 9 wherein the first set of elements includes spare blocks.

11. The method of claim 9 wherein obtaining the erase counts includes obtaining the erase counts from an erase count block.

12. The method of claim 9 wherein the flash memory is NAND flash memory.

13. A memory system comprising:

a first memory, the first memory being arranged to store a table, the table being arranged to include entries associated with a first set of erased storage elements which have associated erase counters that are less than an average erase count associated with the memory system, the associated erase counter for each storage element of the first set of storage elements being arranged to substantially indicate a number of times the storage element has been erased;

a second memory, the second memory being arranged to include a plurality of erased storage elements, the plurality of storage elements including the first set of storage elements, wherein the average erase count is determined using erase counters associated with the plurality of storage elements; and

a processor, the processor being arranged to access the first memory and the second memory.

14. The memory system of claim 13 wherein the first set of storage elements is substantially identified using the table.

15. The memory system of claim 13 wherein the second memory is a flash memory and the first set of storage elements include a first set of blocks.

16. A method for processing elements included in a non-volatile memory of a memory system, the method comprising:

BNSDOCID: <WO_ 2004040459A1_I_>

identifying a first set of elements, wherein the erased elements included in the first set of elements are less worn than the erased elements not included in the first set of elements;

placing entries associated with the first set of elements into a data structure;

sorting the entries within the data structure; and

identifying a first element within the first set of elements using the sorted entries, wherein the first element is less worn than other elements included in the first set of elements.

17. The method of claim 16 wherein each element has an associated erase count, and identifying the first set of elements includes:

identifying a number of erased elements for which the associated erase counts are relatively low; and

grouping the number of erased elements for which the associated erase counts are relatively low into the first set.

18. The method of claim 17 wherein the entries include the associated erase counts of the elements in the first set.

19. The method of claim 17 further including:

determining when an associated erase count of a second element not included in is to be erased;

removing the first element from the first set when it is determined that the second element is to be erased;

copying one of contents of the second element and new contents to replace the contents of the second element into the first element;

erasing the contents from the second element; and

adding the second element to the first set.

20. The method of claim 19 wherein removing the first element from the first set includes removing an entry associated with the first element from the data structure, and adding the second element to the first set includes placing an entry associated with the first set of elements into a data structure.

21. The method of claim 20 further including:

resorting the entries within the data structure; and

identifying a third element within the first set of elements using the sorted entries, wherein the third element is less worn than other elements included in the first set of elements.

BNSDOCID: <WO 2004040459A1 >

22. The method of claim 16 wherein the first element is arranged to replace a second element not included in the first set is erased.

23. The method of claim 16 wherein the elements each include a section that is arranged to contain data, and wherein the section included in each of the elements included in the first set of elements is substantially empty.

24. The method of claim 16 wherein each of the elements included in the first set of elements is substantially erased.

25. The method of claim 16 wherein the non-volatile memory is a flash memory, and the elements are blocks.

26. A memory system comprising:

memory elements;

a system memory component;

means for identifying a first set of the erased memory elements, wherein the memory elements included in the first set of memory elements are less worn than the erased memory elements not included in the first set of memory elements;

means for placing entries associated with the first set of memory elements into a data structure associated with the system memory component;

means for sorting the entries within the data structure; and

means identifying a first memory element within the first set of memory elements using the sorted entries, wherein the first memory element is less worn than other erased memory elements included in the first set of memory elements.

27. The memory system of claim 26 wherein each memory element has an associated erase count, and the means for identifying the first set of elements includes:

means for identifying a number of elements for which the associated erase counts are relatively low; and

means for grouping the number of elements for which the associated erase counts are relatively low into the first set.

28. The memory system of claim 27 wherein the entries include the associated erase counts of the memory elements in the first set.

ENSDOCID: <WO 2004040459A1 I >

29. The memory system of claim 27 further including:

means for determining a first average erase count, the first average erase count being determined using the associated erase count of each memory element, wherein the means for identifying the first set of memory elements include means for comparing the associated erase count of each memory element included in the first set of memory elements to the first average erase count.

30. The memory system of claim 26 further including:

means for removing the first memory element from the first set; and

means for adding a second memory element not included in the first set to the first set.

31. The memory system of claim 30 wherein the means for removing the first memory element from the first set include means for removing an entry associated with the first memory element from the data structure, and the means for adding the second memory element to the first set include means for placing an entry associated with the first set of memory elements into a data structure.

32. The memory system of claim 26 wherein the memory elements included in the first set of memory elements are spare memory elements.

33. The memory system of claim 26 wherein each of the memory elements included in the first set of memory elements is substantially erased.

34. The memory system of claim 26 wherein the memory elements are non-volatile memory blocks.

35. A method for processing a set of memory elements associated with a non-volatile memory system, the method comprising:

grouping at least one of the erased memory elements of the set of memory elements into a first group, wherein the at least one of the memory elements has been substantially no more than memory elements from the set of memory elements which are not grouped into the first group; and

enabling the memory elements grouped into the first group to be allocated for use, wherein enabling the memory elements grouped into the first group to be allocated for use

BNSDOCID: <WO 2004040459A1_I_>

includes substantially preventing the memory elements grouped into the first group from being allocated for use until a first memory element not included in the first group includes contents which are to be disassociated from the first memory element .

36. The method of claim 35 wherein each memory element of the set of memory elements has an associated indication of a number of times the memory element of the set of memory elements has been erased, and grouping the at least one of the memory elements into the first group includes grouping the at least one of the memory elements into the first group based on the associated indication of the at least one of the memory elements.

37. The method of claim 36 wherein the at least one of the memory elements has an associated indication that is less than the associated indications of the memory elements which are not grouped into the first group.

38. A memory management system comprising:

code devices for obtaining erase counts associated with a plurality of elements, wherein each element included in the plurality of elements has an associated erase count, the associated erase count of each element being arranged to indicate a number of times the element has been erased;

code devices for grouping a number of elements included in the plurality of elements into a first set, wherein the code devices for grouping the number of elements into the first set includes code devices for selecting erased elements included in the plurality of elements which have the lowest associated erase counts of the erase counts associated with the plurality of erased elements; and

code devices for storing the erase counts associated with the first set substantially into a table in a memory component.

39. The memory management system of claim 38 wherein the code devices for grouping the number of elements includes:

code devices for comparing the erase counts associated with the plurality of elements; and

code devices for identifying a predetermined number of elements to be the number of elements selected from the plurality of erased elements which have the lowest associated erase counts of the erase counts associated with the plurality of elements.

40. The memory management system of claim 38 further including:

BNSDOCID :WO 2004040459A1 I >

code devices for sorting the elements in the first set according to the erase count associated with each element.

41. The memory management system of claim 40 wherein the code devices for sorting the elements in the first set include code devices for sorting the erase counts within the table.

42. The memory management system of claim 40 further including:

code devices for identifying a first element included in the first set, the first element having the lowest erase count associated with the first set.

43. The memory management system of claim 42 further including:

code devices for disassociating the first element from the first set;

code devices for associating a second element of the plurality of elements with the first set; and

code devices for disassociating the second element from the second set.

44. The memory management system of claim 38 wherein the non-volatile memory is flash memory and the plurality of elements are blocks, the first set being a set of least frequently erased blocks.

45. The memory management system of claim 44 wherein the first set of elements includes spare blocks.

45. The memory management system of claim 44 wherein obtaining the erase counts includes obtaining the erase counts from an erase count block.

46. A method for managing memory, the memory including a plurality of blocks, the method comprising:

identifying a set of erased blocks included in the plurality of blocks;

identifying a first group of erased blocks included in the set of erased blocks, wherein the first group of erased blocks includes erased blocks with lower erase counts than substantially all other erased blocks in the set of erased blocks;

creating a structure in a memory component, wherein the structure includes entries, the entries including erase counts of the erased blocks included in the first group;

sorting the entries; and

BNSDOCID. <WO 2004040459A1 I_>

identifying a first block using the sorted entries, wherein the first block is arranged to be removed from the first group of erased blocks prior to the other erased blocks included in the first group of erased blocks.

47. The method of claim 46 wherein sorting the entries includes sorting the entries using the erase counts of the erased blocks included in the first group.

48. The method of claim 47 wherein identifying the first block using the sorted entries includes identifying the first block to have a lower erase count than substantially all other erased blocks included in the first group.

49. The method of claim 46 wherein the erased blocks are erased non-volatile memory blocks, and the memory component is a NAND memory.

BNSDOCID: <WO_2004040459A1_I_>

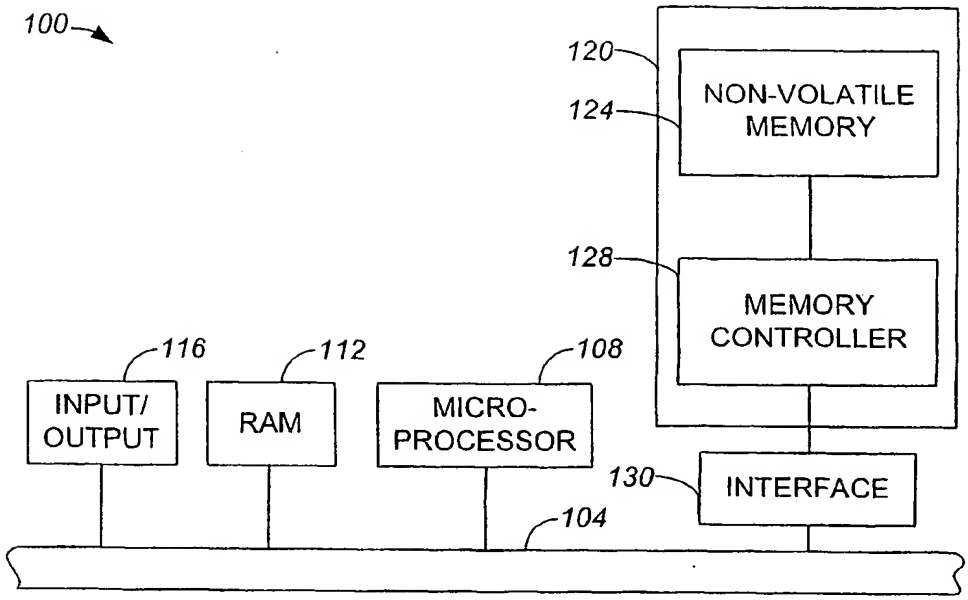

100
120
NON-VOLATILE MEMORY
124
128
MEMORY CONTROLLER
116
112
108
INPUT/ OUTPUT
RAM
MICRO-PROCESSOR
130
INTERFACE
104

Fig. 1a

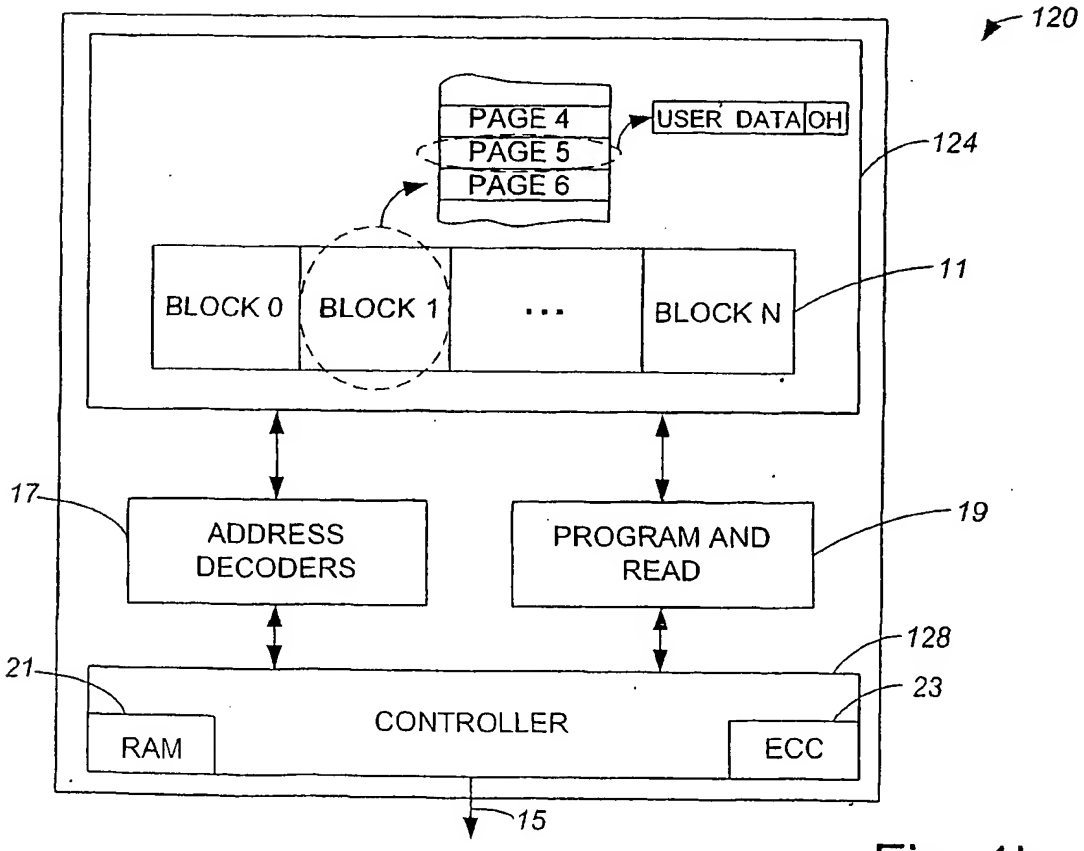

120
PAGE 4
PAGE 5
PAGE 6
USER DATA
OH
124
BLOCK 0
BLOCK 1
...
BLOCK N
11
17
ADDRESS DECODERS
PROGRAM AND READ
19
128
21
23
RAM
CONTROLLER
ECC
15

Fig. 1b

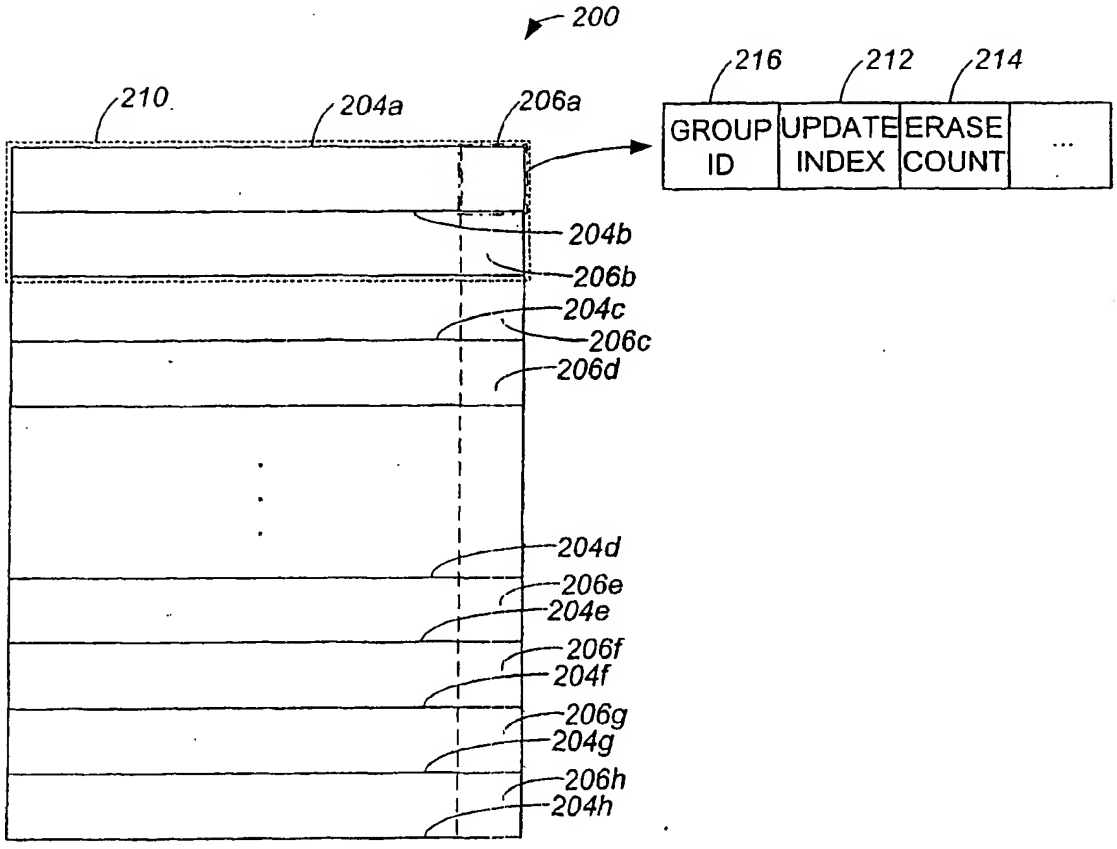

200
216
212
214
210
204a
206a
GROUP ID
UPDATE INDEX
ERASE COUNT
...
204b
206b
204c
206c
206d
204d
206e
204e
206f
204f
206g
204g
206h
204h

Fig. 2

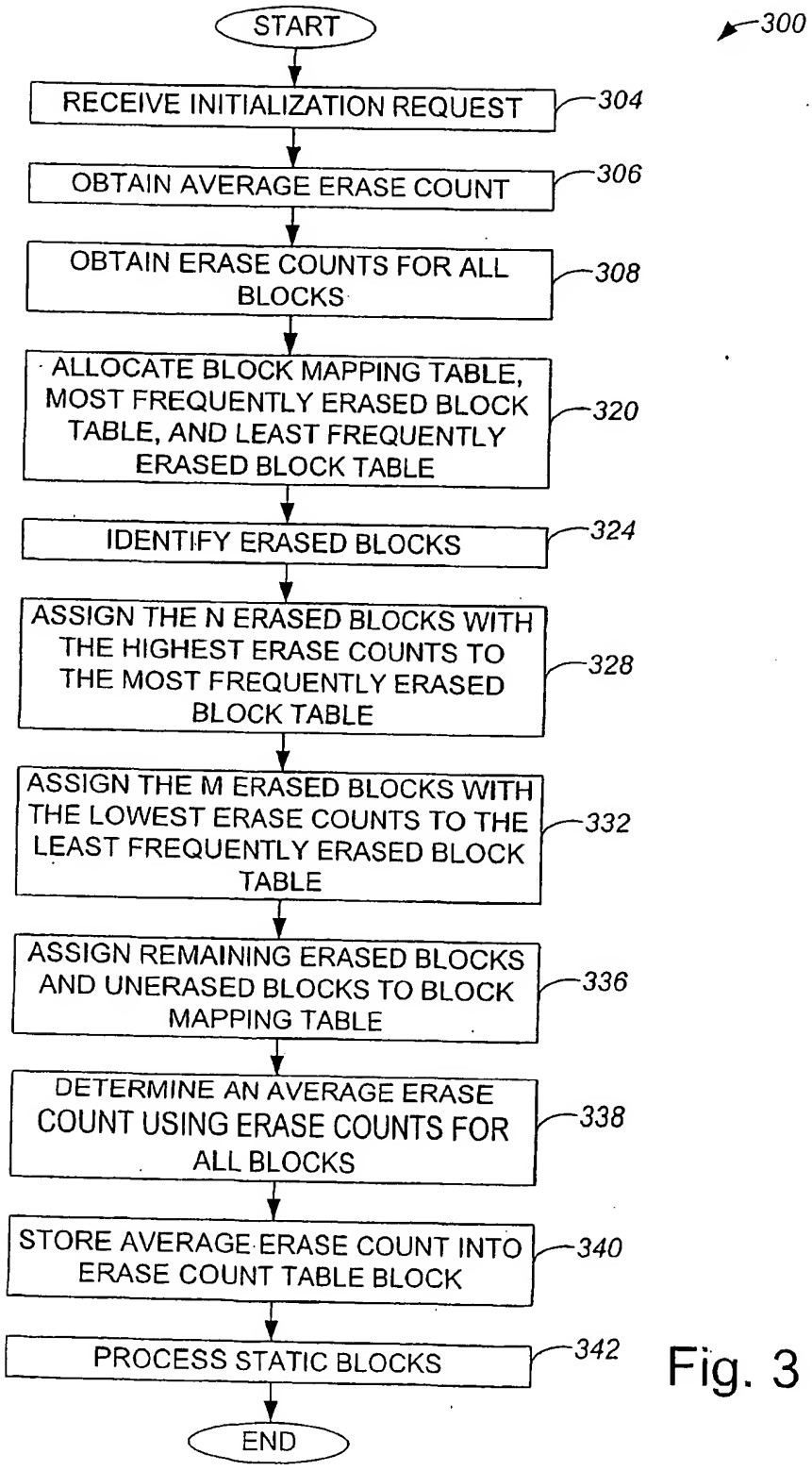

START
300
RECEIVE INITIALIZATION REQUEST
304
OBTAIN AVERAGE ERASE COUNT
306
OBTAIN ERASE COUNTS FOR ALL BLOCKS
308
ALLOCATE BLOCK MAPPING TABLE, MOST FREQUENTLY ERASED BLOCK TABLE, AND LEAST FREQUENTLY ERASED BLOCK TABLE
320
IDENTIFY ERASED BLOCKS
324
ASSIGN THE N ERASED BLOCKS WITH THE HIGHEST ERASE COUNTS TO THE MOST FREQUENTLY ERASED BLOCK TABLE
328
ASSIGN THE M ERASED BLOCKS WITH THE LOWEST ERASE COUNTS TO THE LEAST FREQUENTLY ERASED BLOCK TABLE
332
ASSIGN REMAINING ERASED BLOCKS AND UNERASED BLOCKS TO BLOCK MAPPING TABLE
336
DETERMINE AN AVERAGE ERASE COUNT USING ERASE COUNTS FOR ALL BLOCKS
338
STORE AVERAGE ERASE COUNT INTO ERASE COUNT TABLE BLOCK
340
PROCESS STATIC BLOCKS
342
END

Fig. 3

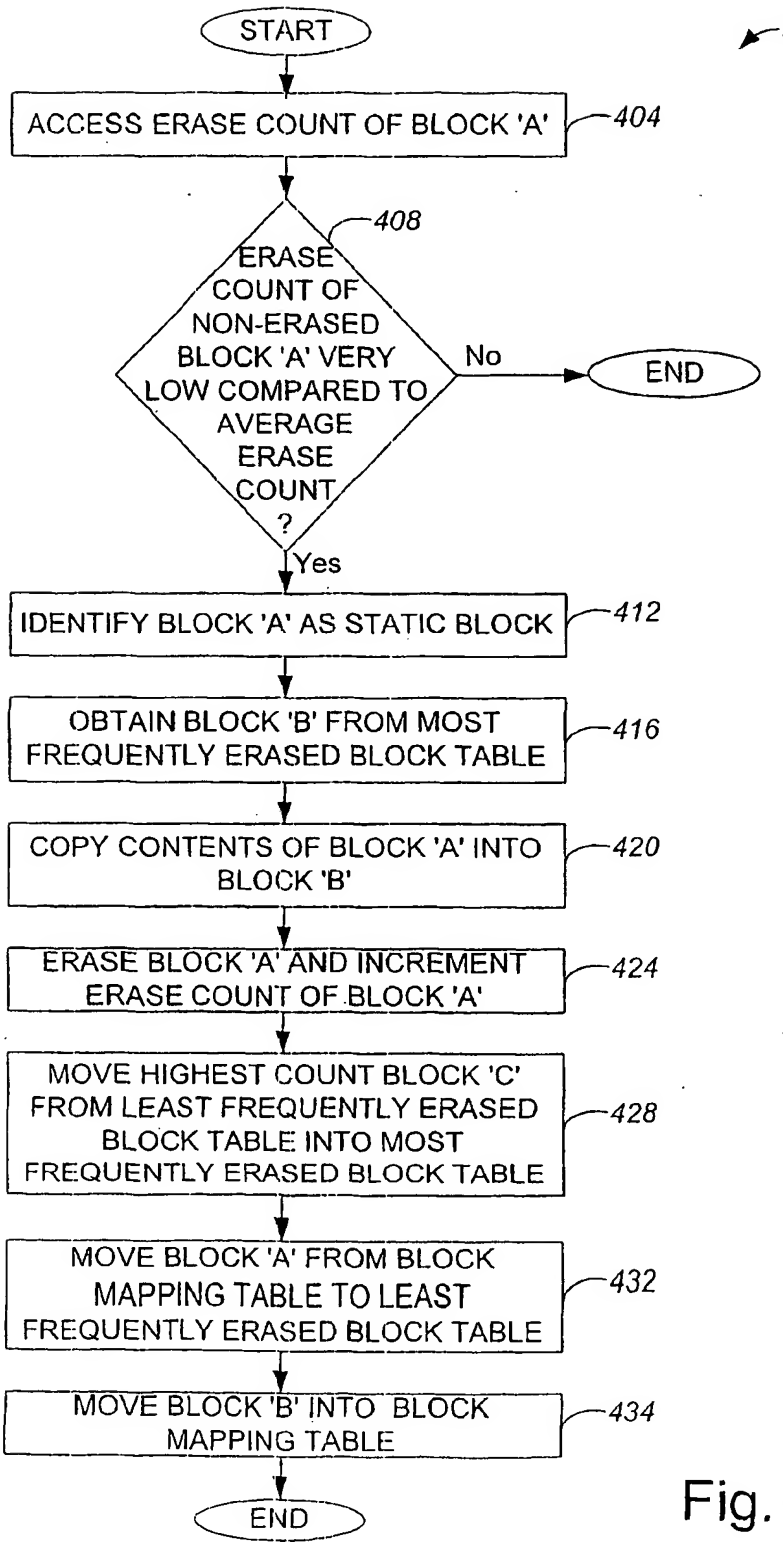

342
START
ACCESS ERASE COUNT OF BLOCK 'A'
404
408
ERASE COUNT OF NON-ERASED BLOCK 'A' VERY LOW COMPARED TO AVERAGE ERASE COUNT ?
No
END
Yes
IDENTIFY BLOCK 'A' AS STATIC BLOCK
412
OBTAIN BLOCK 'B' FROM MOST FREQUENTLY ERASED BLOCK TABLE
416
COPY CONTENTS OF BLOCK 'A' INTO BLOCK 'B'
420
ERASE BLOCK 'A' AND INCREMENT ERASE COUNT OF BLOCK 'A'
424
MOVE HIGHEST COUNT BLOCK 'C' FROM LEAST FREQUENTLY ERASED BLOCK TABLE INTO MOST FREQUENTLY ERASED BLOCK TABLE
428
MOVE BLOCK 'A' FROM BLOCK MAPPING TABLE TO LEAST FREQUENTLY ERASED BLOCK TABLE
432
MOVE BLOCK 'B' INTO BLOCK MAPPING TABLE
434
END

Fig. 4

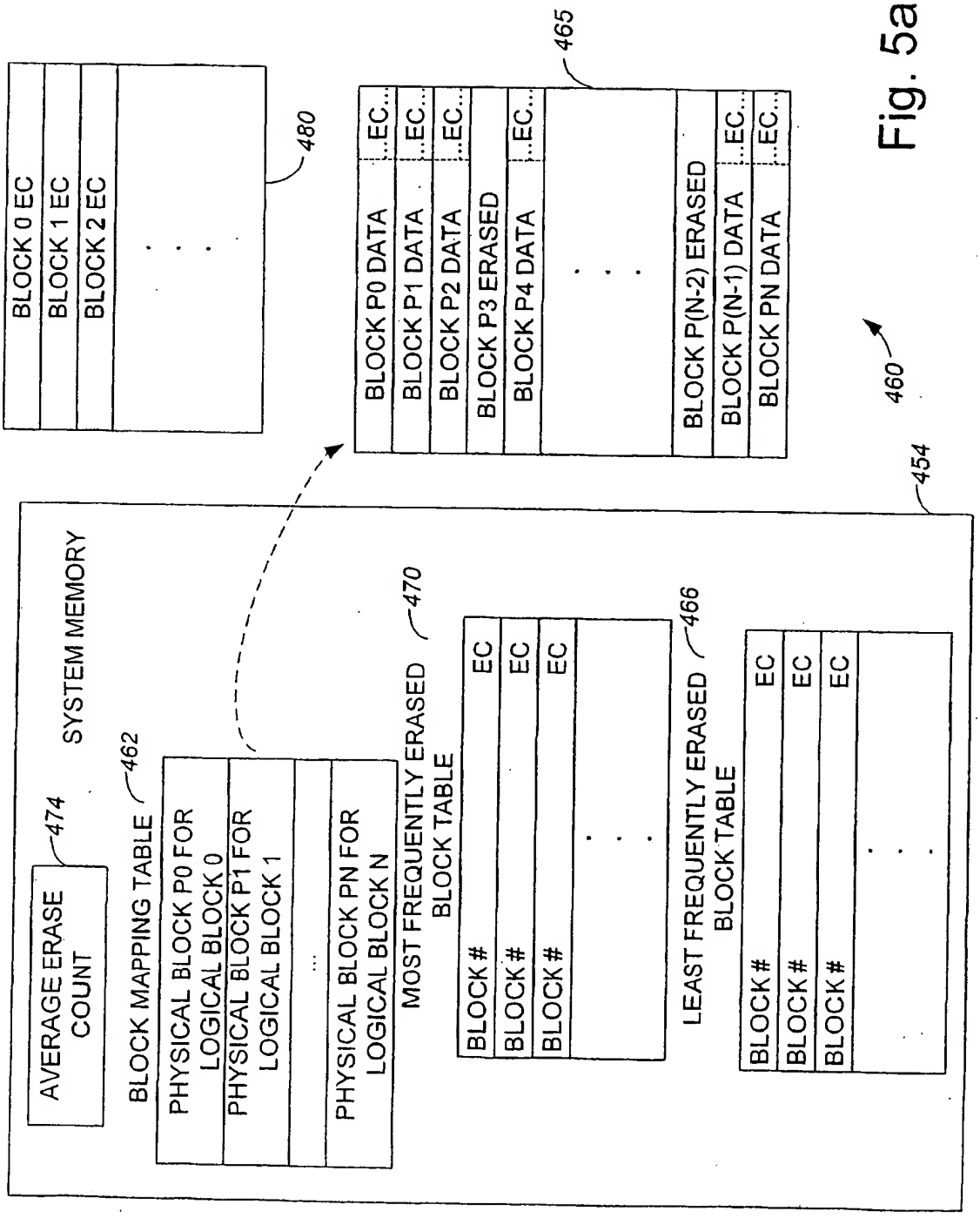

BLOCK 0 EC
BLOCK 1 EC
BLOCK 2 EC
480
BLOCK P0 DATA ...EC...
BLOCK P1 DATA ...EC...
BLOCK P2 DATA ...EC...
BLOCK P3 ERASED
BLOCK P4 DATA ...EC...
BLOCK P(N-2) ERASED
BLOCK P(N-1) DATA ...EC...
BLOCK PN DATA ...EC...
465
460
454
SYSTEM MEMORY
AVERAGE ERASE COUNT
474
BLOCK MAPPING TABLE
462
PHYSICAL BLOCK P0 FOR LOGICAL BLOCK 0
PHYSICAL BLOCK P1 FOR LOGICAL BLOCK 1
PHYSICAL BLOCK PN FOR LOGICAL BLOCK N
MOST FREQUENTLY ERASED BLOCK TABLE
470
BLOCK # EC
BLOCK # EC
BLOCK # EC
LEAST FREQUENTLY ERASED BLOCK TABLE
466
BLOCK # EC
BLOCK # EC
BLOCK # EC

Fig. 5a

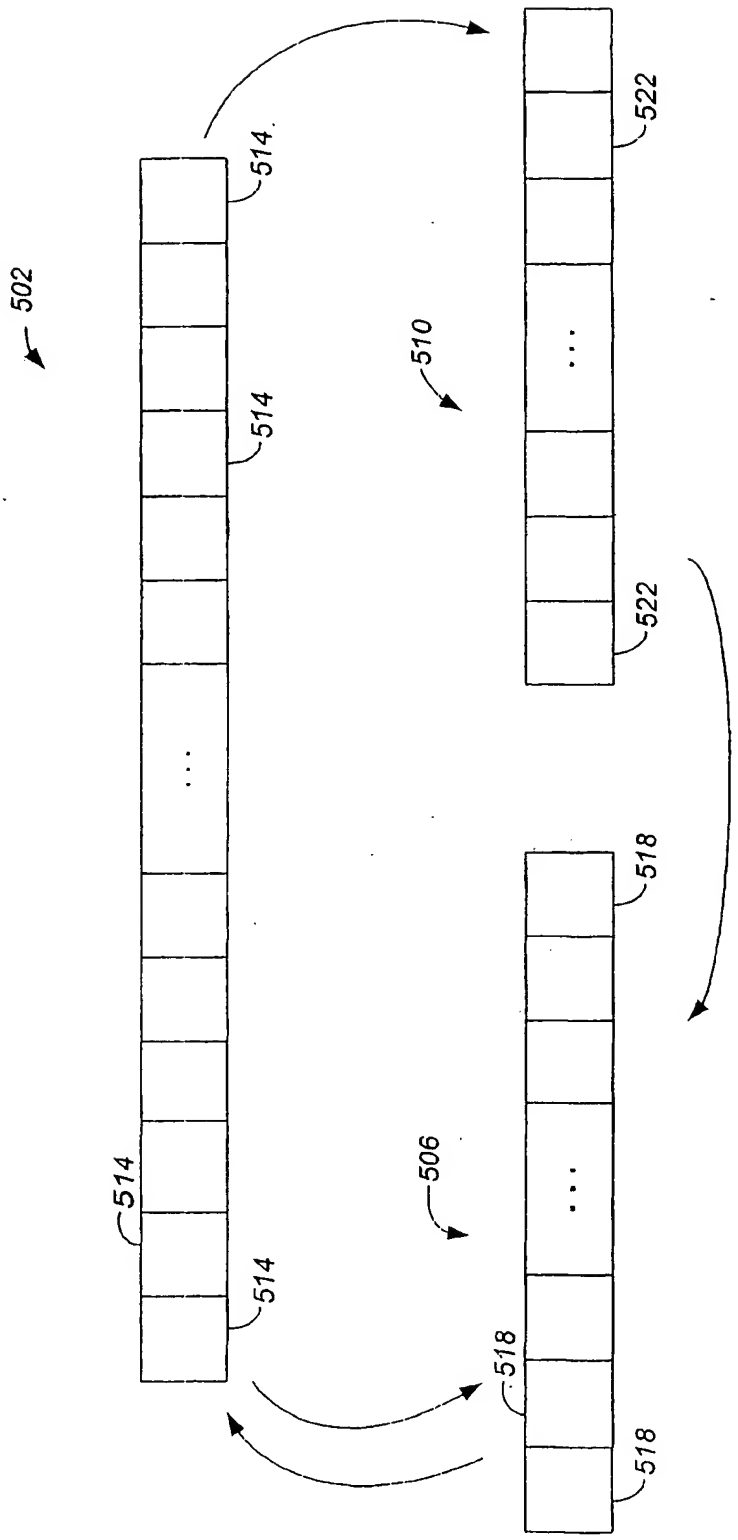


Fig. 5b

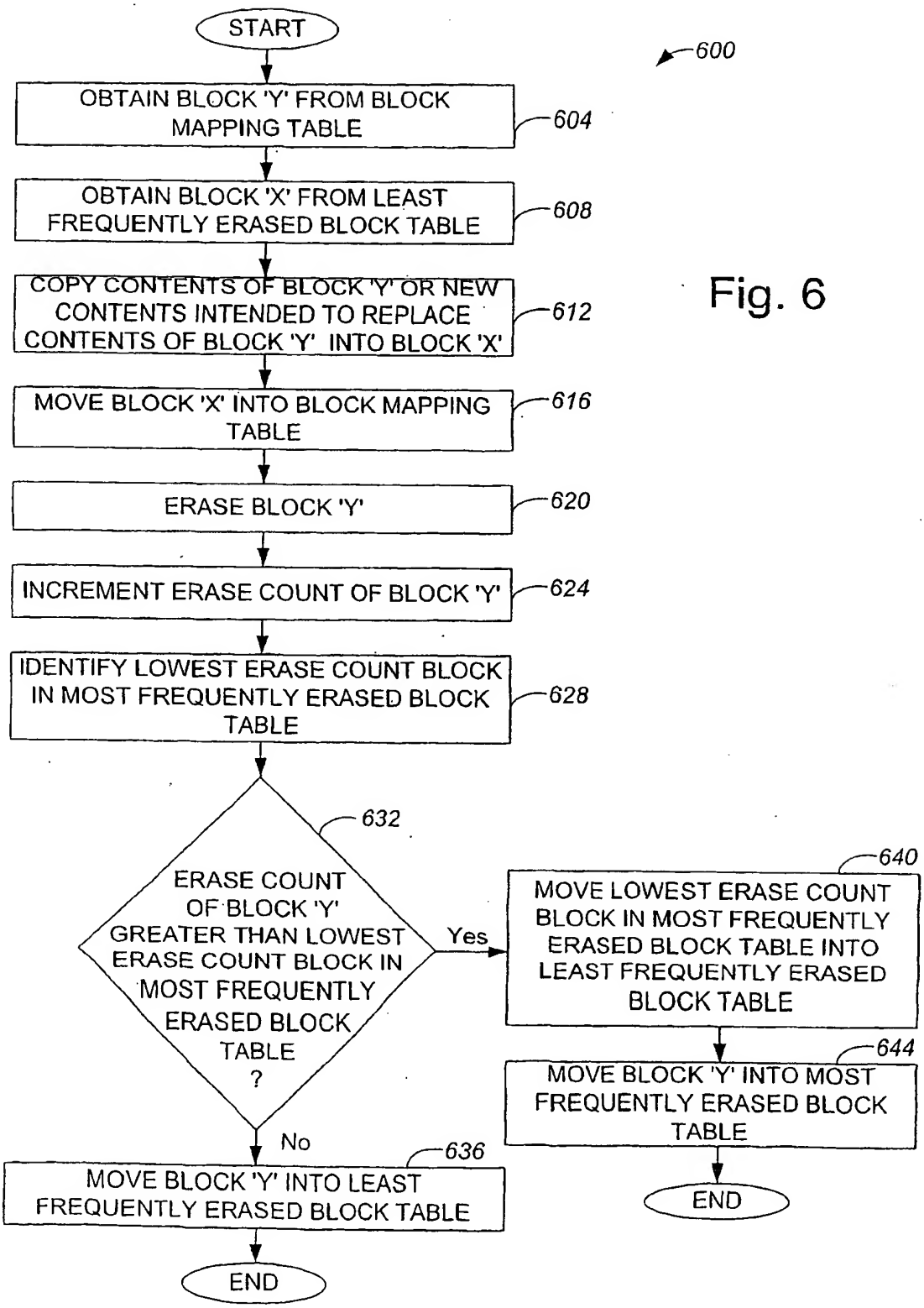

START
600
OBTAIN BLOCK 'Y' FROM BLOCK MAPPING TABLE
604
OBTAIN BLOCK 'X' FROM LEAST FREQUENTLY ERASED BLOCK TABLE
608
COPY CONTENTS OF BLOCK 'Y' OR NEW CONTENTS INTENDED TO REPLACE CONTENTS OF BLOCK 'Y' INTO BLOCK 'X'
612
MOVE BLOCK 'X' INTO BLOCK MAPPING TABLE
616
ERASE BLOCK 'Y'
620
INCREMENT ERASE COUNT OF BLOCK 'Y'
624
IDENTIFY LOWEST ERASE COUNT BLOCK IN MOST FREQUENTLY ERASED BLOCK TABLE
628
632
ERASE COUNT OF BLOCK 'Y' GREATER THAN LOWEST ERASE COUNT BLOCK IN MOST FREQUENTLY ERASED BLOCK TABLE ?
Yes
640
MOVE LOWEST ERASE COUNT BLOCK IN MOST FREQUENTLY ERASED BLOCK TABLE INTO LEAST FREQUENTLY ERASED BLOCK TABLE
644
MOVE BLOCK 'Y' INTO MOST FREQUENTLY ERASED BLOCK TABLE
END
No
636
MOVE BLOCK 'Y' INTO LEAST FREQUENTLY ERASED BLOCK TABLE
END

Fig. 6

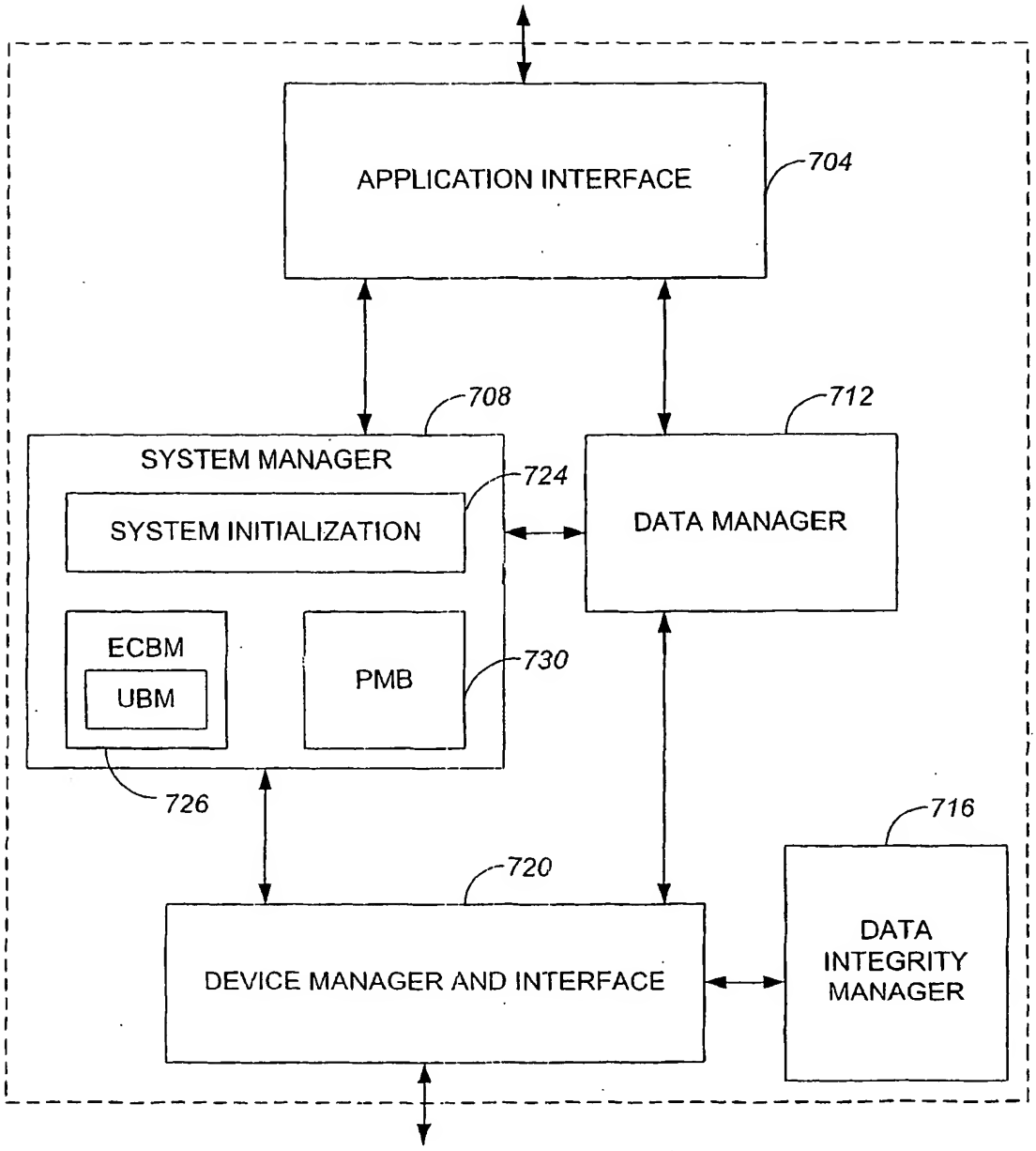

APPLICATION INTERFACE
704
708
712
SYSTEM MANAGER
724
SYSTEM INITIALIZATION
DATA MANAGER
ECBM
UBM
PMB
730
726
716
720
DEVICE MANAGER AND INTERFACE
DATA INTEGRITY MANAGER
700

Fig. 7

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 03/28429

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F12/02

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F G11C

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 568 423 A (JOU EDWIN ET AL) 22 October 1996 (1996-10-22) column 3, line 49 -column 4, line 26; figures 1,2A,3,4	1,12,16, 18,24, 26-28, 33, 35-38, 46,49
X	column 3, line 49 -column 4, line 26; figure 2A	2-4,16, 26, 39-41,47
X	column 3, line 12,13	5,17,26, 42,48
X	column 6, line 26-36 column 4, line 27 -column 5, line 18; figure 2B column 5, line 19-41	6,7, 19-22, 30,31,43

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 March 2004	19/03/2004
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Weber, R

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 03/28429

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	figure 1	8,9,25, 34,44
X	column 3, line 58-67; figure 2A	10,32,45
X	column 3, line 58,59; figure 2A	11,46
X	US 5 930 193 A (ACHIWA KYOSUKE ET AL) 27 July 1999 (1999-07-27) column 4, line 9-56; figures 1,2 column 6, line 13-22 column 6, line 50-57	13-15,29

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

BNSDOCID: <WO 2004040459A1 >

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 03/28429

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5568423 A	22-10-1996	NONE	
US 5930193 A	27-07-1999	JP 8016482 A	19-01-1996
		US 5737742 A	07-04-1998

Form PCT/ISA/210 (patent family annex) (July 1992)

BNSDOCID <WO 2004040459A1 >